AUTODESK

Autodesk AEC Collection

Dynamo トレーニングテキスト　活用術 2

Dynamo for Revit を使って橋台に配筋する
(Dynamo for Civil 3D, Dynamo for Revit)

2020 年 8 月 24 日
Ver 1.0

目　　次

1. Dynamo for Revit を使って橋台に配筋する

a. はじめに

初級編と活用術 1 では、「〇〇というノードを使えば△△ができます」という説明を、一つ一つ丁寧にしてきました。しかし、本当に Dynamo を使って自分のしたいことを行うには、このテキストで説明しきれていない膨大なノードやパッケージを組み合わせ、自分でプログラムを組み立てる必要があります。
上級編では「Excel シートを読み取って、Revit で作った橋台モデルに配筋する」という Dynamo を作成します。これは、別にあなたのしたいことではない可能性が高いでしょう。しかし、これを実現するための「ものの考え方」「使えそうな情報源」を知っておけば、同じような段取りで、あなたのしたいことを実現できるでしょう。ということで、Dynamo の操作に入る前に、本当に Dynamo を使いこなすための「心構え」をお伝えします。

- 論理的に考える：今回のプログラムの目的は「Excel シートを読み取って、Revit で作った橋台モデルに配筋する」ことです。これを、どういう順番で実現するか整理するのが、最初のステップです。今回の例で言うと、 以下のようになるでしょうか。
    - プログラム全体の入力値は？ ： Excel シートの値、Revit の橋台モデル
    - プログラム全体の出力値は？ ： 鉄筋
    - 入力値から出力値まで、どのようにつなげるか？ ：
        - 鉄筋ファミリを Dynamo に読み込む
        - 橋台の位置を Dynamo に読み込む
        - 橋台の位置に合わせて、鉄筋の位置を計算する
        - 読み込んだ鉄筋ファミリを、計算した鉄筋の位置に配置する

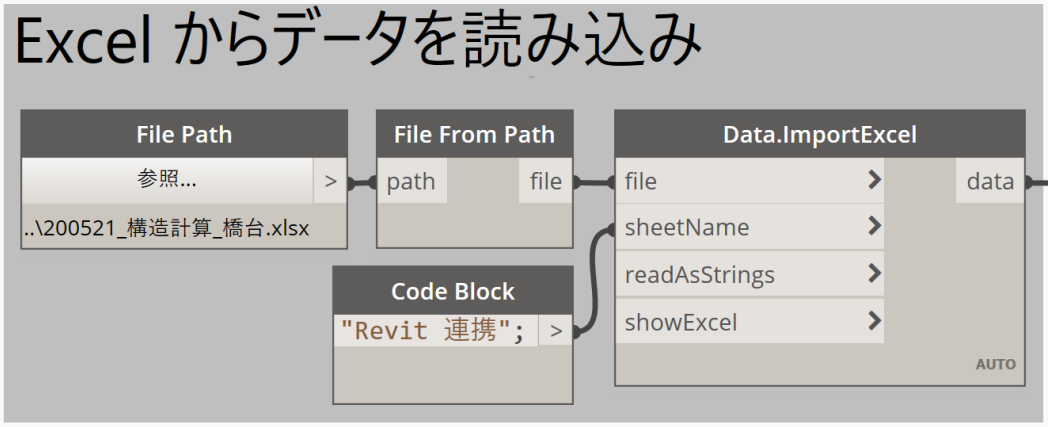
- 英語で調べる：日本語が話せる人は 1 億人しかいませんが、英語が話せる（読める、書ける）人は 20 億人います。単純に考えれば、英語が使えると、Google 検索で 20 倍の情報にアクセスできます。今回使う Dynamo のパッケージも、英語でならドキュメントが提供されています。英語が苦手でも、google translate を使えばあたりは付けられます。「英語で調べる」のは Google 検索に限らず、Dynamo の検索バーでも同じです。例えば、「Dynamo を使って Revit モデルの位置を知りたい」とします。どのノードを使えばいいでしょうか？ この時、「位置」を表す英単語（e.g. position, location …）を Dynamo の検索バーに片っ端から入れていくと、あたりが付けられます。同じように、モデルを平行移動したければ“move”や“translate”などでしょうし、体積を計算したければ“volume”や“capacity”などでしょう。自分のしたいことを実現するには、このような「検索力」がモノを言います。

- 公式ドキュメントやフォーラムを参考にする：Autodesk の他のソフトウェアと同じですが、分からないことを検索すると、公式ドキュメントやフォーラムが引っ掛かることが多いです。調べ物をする際は、以下のリンクを参考にしてみてください。
  https://primer.dynamobim.org/ja/（チュートリアル）
  https://dictionary.dynamobim.com/#（ノードの説明）
  https://forum.dynamobim.com/（様々な質問や回答が掲載されたフォーラム）

では、実際にプログラムを作成していきましょう。最終的なデータは “Add_Straight_Rebar_Footings.dyn” に入っています。以下では、大まかな作業フローのみ紹介しています。
※“Add_Straight_Rebar_Footings.dyn”の読み込み手順は、活用術 1 を参照してください。

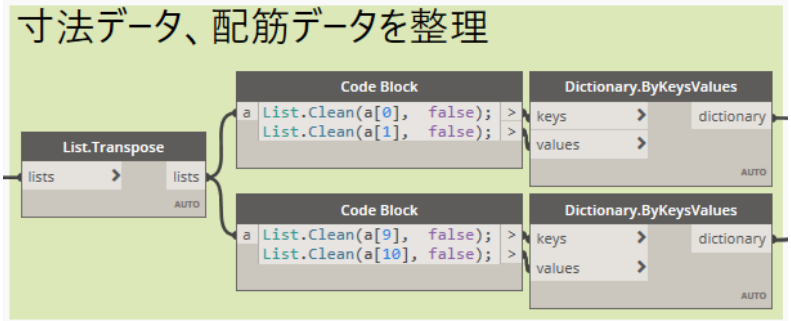
b. Excel データを読み込み

まず、下図のように Excel からデータを読み込むグループを作成します。

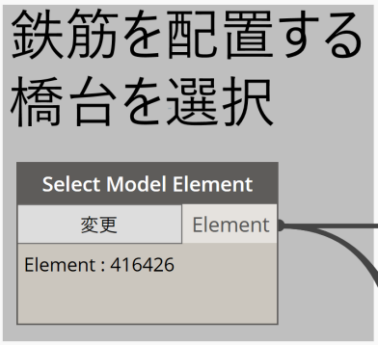


c. 寸法データ、配筋データを整理

Excel データを読み込んだら、以降の処理で使いやすいように整理します。Excel データの行と列を逆にしたのも、寸法や配筋に関するデータをディクショナリ形式で保存したのも、その方が以降の処理がしやすいためです。なぜでしょうか？ 考えてみてください。



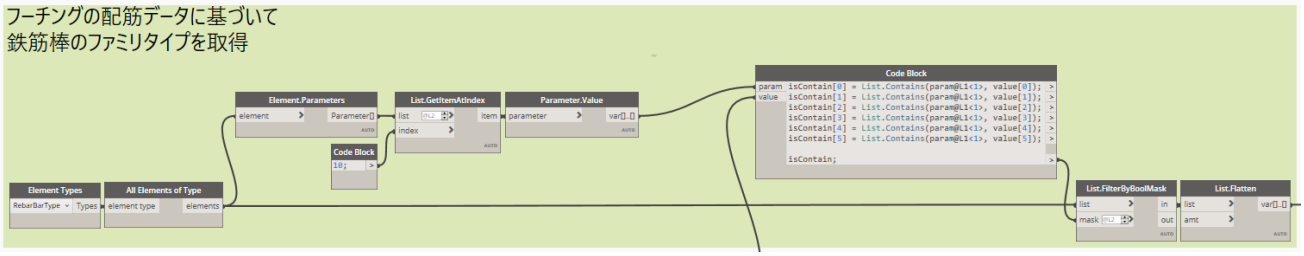
d. 橋台要素を読み込み



今回の入力値は Excel 以外にもあります。鉄筋を配置していく橋台です。

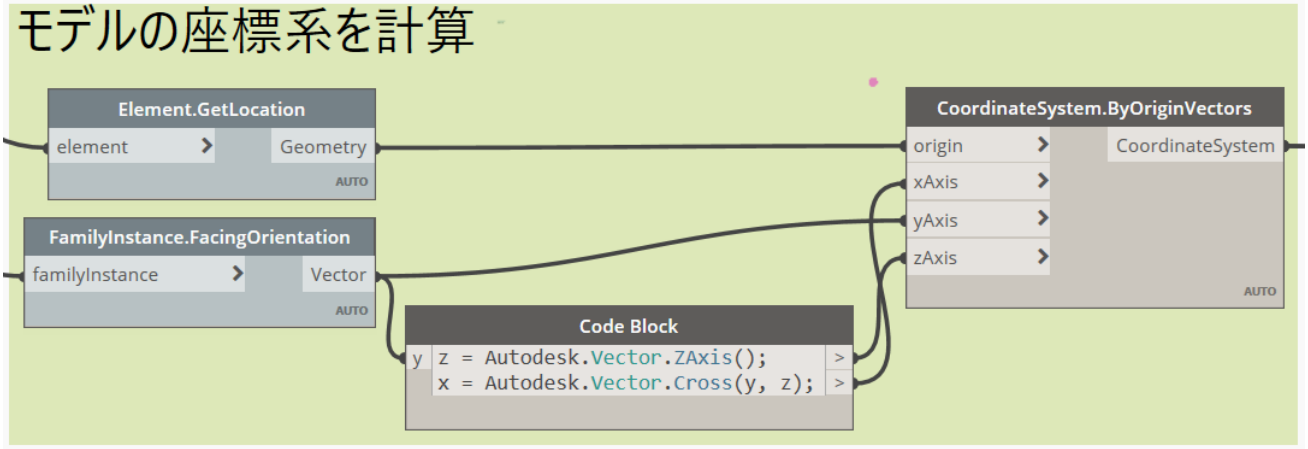
e. 鉄筋棒のファミリタイプを取得

ここからは、活用術 1 で紹介した ”Structural Design” パッケージも使っていきます。”Structural Design” パッケージでは、鉄筋の要素タイプを指定することで、特定の径や曲げ半径を持つ鉄筋を描くことができます。どの要素タイプにすればいいでしょうか？ Excel データから引っ張ってきた鉄筋径と、鉄筋の要素タイプがパラメータとして持つ鉄筋径を照合すれば、どの要素タイプを使用するか、決定できそうです。



f. 橋台の座標系を計算

橋台は、原点に配置されているとも限りませんし、真北を向いているとも限りません。そこで、予め橋台の位置と向きを取得しておき、それを基に座標系を作っておきます。なぜか？ これから行う鉄筋の中心線の計算が、飛躍的に楽になるからです。



g. 鉄筋の中心線を計算

この部分が、このプログラムの本丸です。まずは、鉄筋の中心線の計算をしやすくするために、パラメータを整理しておきます。どのようにパラメータを整理するべきか？ それは、あなたが次の処理で何をしたいかに依ります。

鉄筋の位置と形状を決定するためのパラメータを整理

パラメータの整理

```
dim     dim;
rebar   rebar;

// 主筋 上端
mainUp      = Dictionary.ValueAtKey(rebar, ["主筋 前足 上端 規格", "主筋 後足 上端 規
mainUpD     = String.ToNumber(String.Remove(mainUp@L1<1>, 0, 1));
mainUpCover = Dictionary.ValueAtKey(rebar, ["主筋 前足 上端 かぶり", "主筋 後足 上端
mainUpPitch = Dictionary.ValueAtKey(rebar, ["主筋 前足 上端 ピッチ", "主筋 後足 上端 ピ

// 主筋 下端
mainDown      = Dictionary.ValueAtKey(rebar, ["主筋 前足 下端 規格", "主筋 後足 下端
mainDownD     = String.ToNumber(String.Remove(mainUp@L1<1>, 0, 1));
mainDownCover = Dictionary.ValueAtKey(rebar, ["主筋 前足 下端 かぶり", "主筋 後足 下端
mainDownpitch = Dictionary.ValueAtKey(rebar, ["主筋 前足 下端 ピッチ", "主筋 後足 下端

// 配力筋
dist          = Dictionary.ValueAtKey(rebar, "配力筋 規格");
distD         = String.ToNumber(String.Remove(dist, 0, 1));
distUpCover   = Math.Min(mainUpCover[0]-0.5*(mainUpD[0]+distD), mainUpCover[1]-0.
distDownCover = Math.Min(mainDownCover[0]-0.5*(mainDownD[0]+distD), mainDownCover
distPitch     = Dictionary.ValueAtKey(rebar, "配力筋 ピッチ");

// 躯体の寸法
wallDepth       =   Dictionary.ValueAtKey(dim, "竪壁 厚さ");
footFrontLength =   Dictionary.ValueAtKey(dim, "前足 長さ") + wallDepth;
footBackLength  =   Dictionary.ValueAtKey(dim, "後足 長さ");
footWidth       =   Dictionary.ValueAtKey(dim, "奥行");
footHeight      =   Dictionary.ValueAtKey(dim, "フーチング 厚さ");

// 原点からのずれ
dz              = -(Dictionary.ValueAtKey(dim, "土被り") + footHeight);
```

パラメータを整理したら、実際に鉄筋の中心線を計算します。鉄筋の径やかぶり、躯体の寸法は整理済みなので、それを踏まえ、鉄筋の端点や屈曲点の座標 (x, y, z) を求めていきます。一本の鉄筋で座標が求まれば、あとはピッチ間隔で複写すれば完了です。これを、主筋（フーチング前足）、主筋（フーチング後足）、配力筋、の三種類で行います。

主筋（後足）の中心線を計算

主筋 後足 中心線を計算

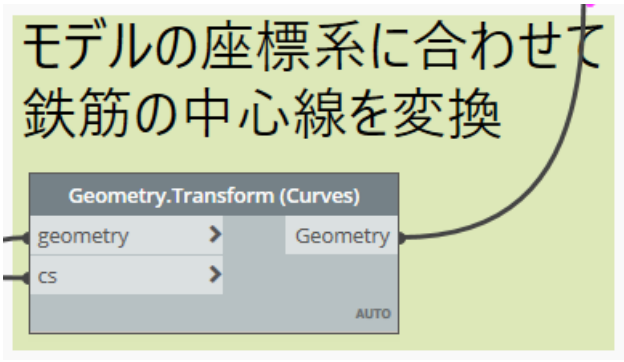
```
mainUpD
mainUpCover
mainUpPitch
mainDownD
mainDownCover
mainDownPitch
wallDepth
footBackLength
footWidth
footHeight
dz
[mainUpD, mainUpCover, mainUpPitch];
[mainDownD, mainDownCover, mainDownPitch];
[wallDepth, footBackLength, footWidth, footHeight];
dz;

dy = Math.Max(0.5*(mainUpD[0]+mainUpD[1]), 0.5*(mainDownD[0]+mainDownD[1]));

// 上端筋の端点 (y=0)
pUp[0] = Point.ByCoordinates(-(footBackLength-mainDownCover[1]-0.5*(mainUpD[1]+ma
pUp[1] = Point.ByCoordinates(-(footBackLength-mainDownCover[1]-0.5*(mainUpD[1]+ma
pUp[2] = Point.ByCoordinates(  wallDepth-mainUpCover[1],

// 上端筋の中心線 (y=0)
lUp[0] = Line.ByStartPointEndPoint(pUp[0], pUp[1]);
lUp[1] = Line.ByStartPointEndPoint(pUp[1], pUp[2]);
lUp;

// 下端筋の端点 (y=0)
pDown[0] = Point.ByCoordinates(-(footBackLength-mainDownCover[1]), dy, dz+footHei
pDown[1] = Point.ByCoordinates(-(footBackLength-mainDownCover[1]), dy, dz
pDown[2] = Point.ByCoordinates(  wallDepth-mainDownCover[1],       dy, dz

// 下端筋の中心線 (y=0)
lDown[0] = Line.ByStartPointEndPoint(pDown[0], pDown[1]);
lDown[1] = Line.ByStartPointEndPoint(pDown[1], pDown[2]);
lDown;

// y 座標
yMin = [-(0.5*footWidth-mainUpCover[1]), -(0.5*footWidth-mainDownCover[1])];
yMax = [  0.5*footWidth-mainUpCover[1],    0.5*footWidth-mainDownCover[1] ];
yMin[0]..yMax[0]..mainUpPitch[1];
yMin[1]..yMax[1]..mainDownPitch[1];
```

Geometry.Translate
geometry
xTranslation
yTranslation
zTranslation
Geometry
AUTO

Geometry.Translate
geometry
xTranslation
yTranslation
zTranslation
Geometry
AUTO

h. 橋台の座標系に合わせて鉄筋の中心線を変換

先ほど鉄筋の中心線を求めました。しかし、これは「橋台が原点に配置されていて、真北を向いている」前提で計算したものです。実際の橋台は、そのように配置されているとは限りません。そこで、前もって作っておいた橋台の座標系に沿うように、鉄筋の中心線を変換します。これで、橋台がどの位置でも、どの向きでも、鉄筋を正しく配置できるようになります。



i. 鉄筋を出力

鉄筋を出力します。必要な入力値は下記のとおりです。

Curves：鉄筋の中心線（曲線や直線で表現）
RebarStyle：直線（主筋や配力筋）なのか U 型（帯筋やあばら筋）なのか
RebarBarType：鉄筋の規格（D16、D25 など）
StartHookType, EndHookType：フックのタイプ
StartHookOrientation, EndHookOrientation：フックの向き
HostElement：鉄筋を配置するモデル
Vector：法線（この平面上で鉄筋のフックが曲がります）

〒104-6024　東京都中央区晴海 1-8-10
晴海アイランド トリトンスクエア　オフィスタワーX24F